Name: Mohit Kerkar Div: D15C Roll No: 23

Adv DevOps Exp 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services. The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don’t need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We’ll go over this in detail later on in
this guide.

Execution Model

The container (and the resources used by it) that runs our function is managed completely by AWS. It is brought up when an event takes place and is turned off if it is not being used. If additional requests are made while the original event is being served, a new container is brought
up to serve a request. This means that if we are undergoing a usage spike, the cloud provider simply creates multiple instances of the container with our function to serve those requests. This has some interesting implications. Firstly, our functions are effectively stateless. Secondly,

each request (or event) is served by a single instance of a Lambda function. This means that you
are not going to be handling concurrent requests in your code. AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

The above execution model makes Lambda functions effectively stateless. This means that every
time your Lambda function is triggered by an event it is invoked in a completely new environment. You don't have access to the execution context of the previous event.
However, due to the optimization noted above, the actual Lambda function is invoked only once per container instantiation. Recall that our functions are run inside containers. So when a function is first invoked, all the code in our handler function gets executed and the handler function gets invoked. If the container is still available for subsequent requests, your function will get invoked and not the code around it.
For example, the createNewDbConnection method below is called once per container instantiation and not every time the Lambda function is invoked. The myHandler function on the other hand is called on every invocation.

Common Use Cases for Lambda

Due to Lambda's architecture, it can deliver great benefits over traditional cloud computing setups for applications where:
1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application.
Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a single HTTP request. Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to

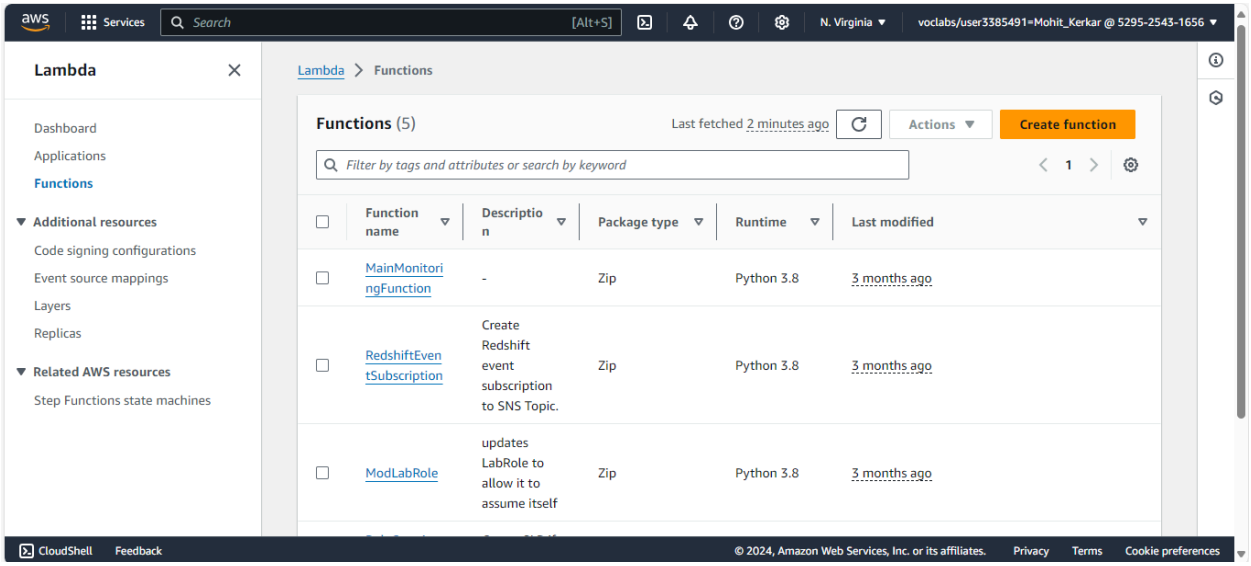
the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.
Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambda function for specific kinds of data events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

Name: Mohit Kerkar Div: D15C Roll No: 23

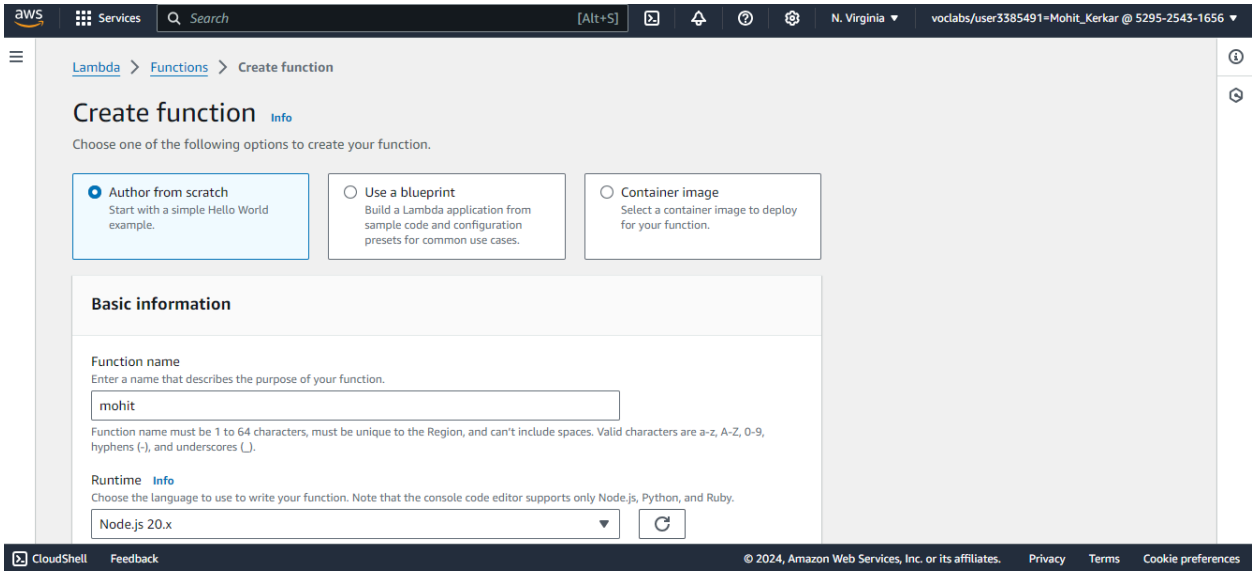
Steps to create and test your lambda function are as follows:-

Step 1: Creating a Function inside lambda

Navigate to Lambda inside AWS services and click on ‘Create Function’ button to create a new function inside Lambda services

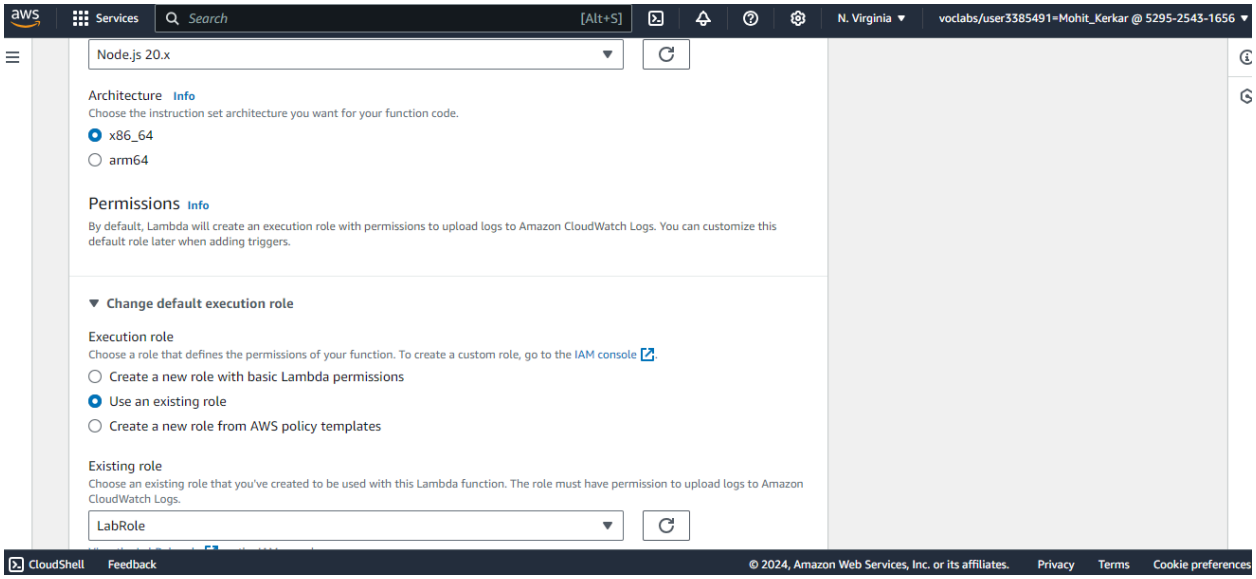


Inside the Create function process, we are supposed to enter the details such as the name of the function, details regarding the things that we to want the function to run or test (eg. programming language).

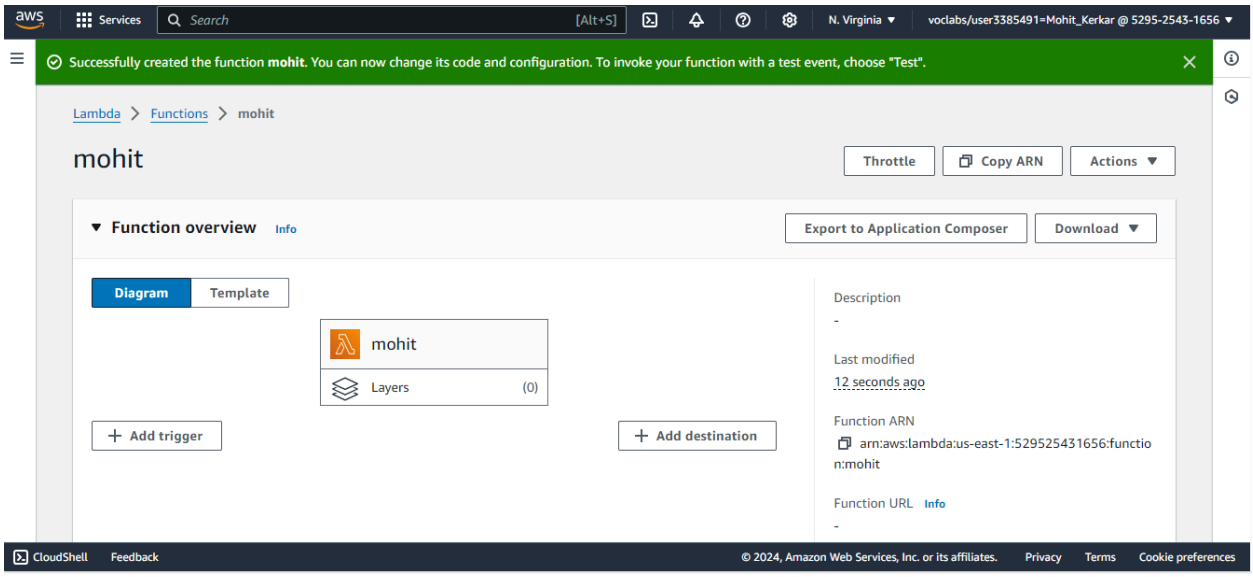


Along with these things, we are also required to set permissions for us to access and use the lambda function efficiently. For that purpose, we are to select ‘Use an existing role’ option and

select ‘LabRole’ which is an already existing role with permissions to use the Lambda Service.



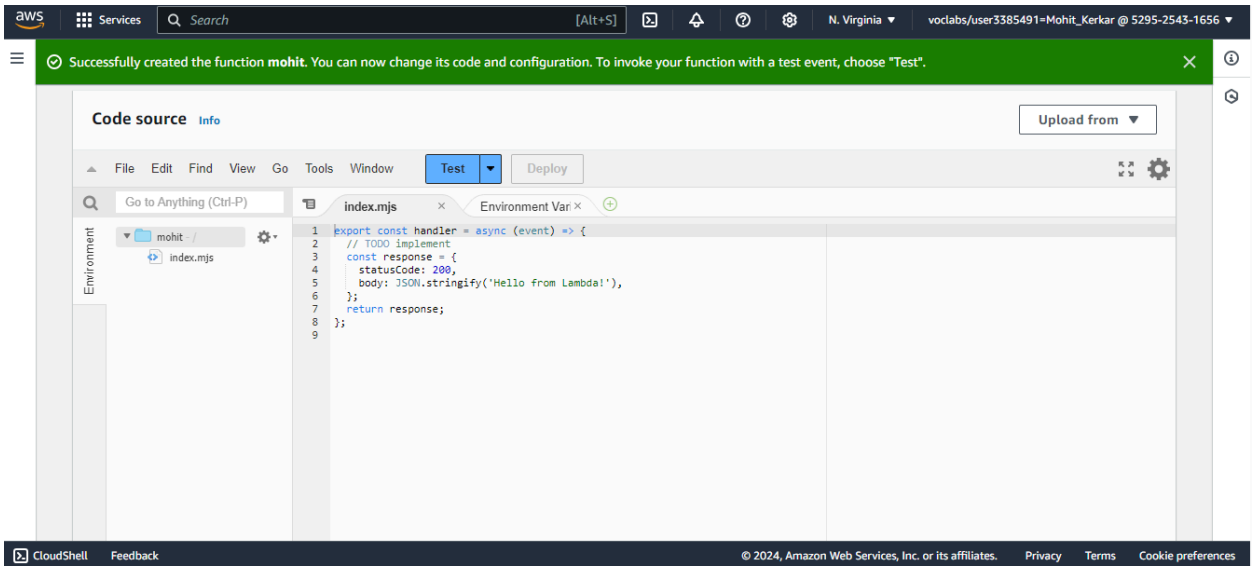
And that is how our function inside Lambda is successfully created. This is what appears on the screen after creation of our function (Our dashboard)



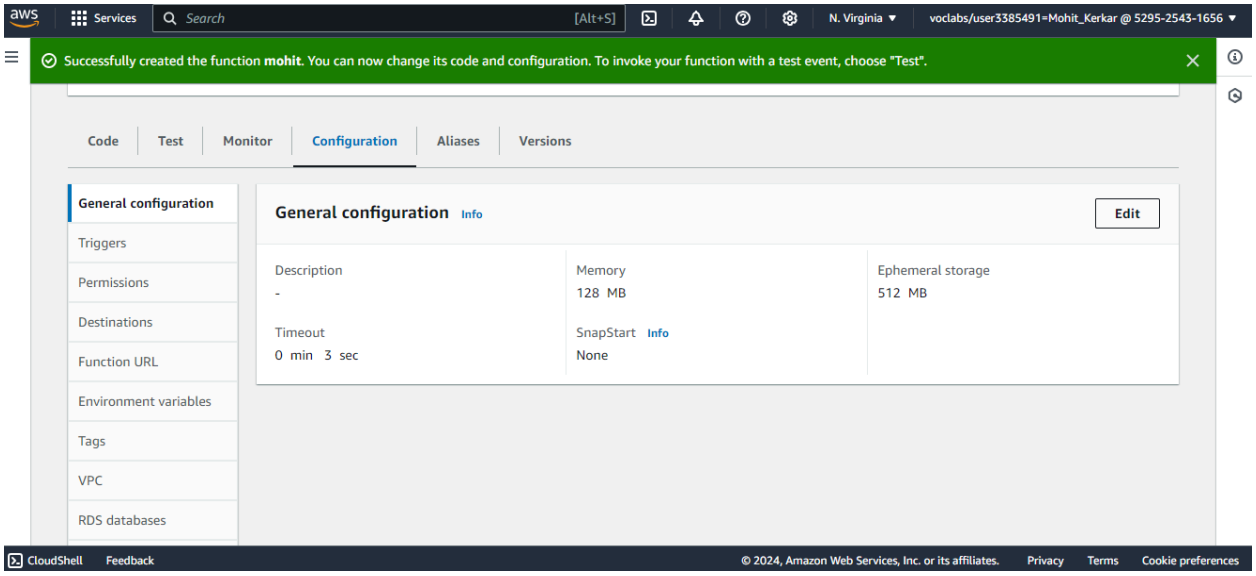
Step 2: Modifying function’s configuration

Name: Mohit Kerkar Div: D15C Roll No: 23

Scroll down to view the default code given in the index.mjs file as we see in the screenshot attached below

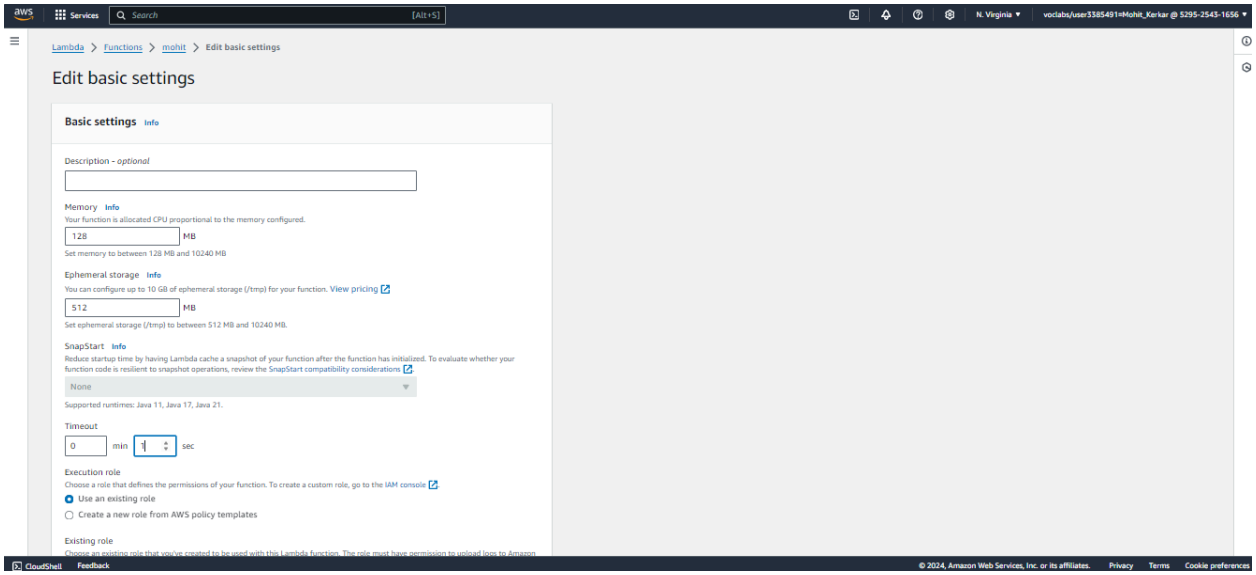


Navigate to the configuration section of our function. This section provides us details about the settings that have been applied to our function for carrying out functionalities like testing on our code in the language that we specified earlier

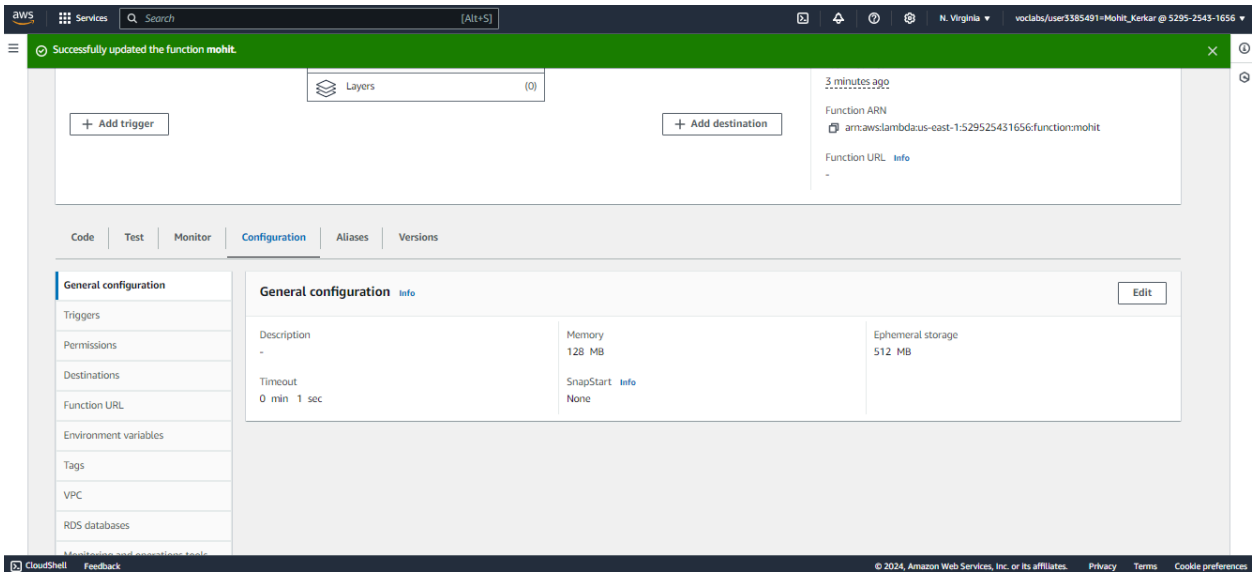


Name: Mohit Kerkar Div: D15C Roll No: 23

Click on Edit. Change the timeout to 1 sec to modify the time for the function to be kept running before forcibly terminating it.
After changing the configuration, click on save.



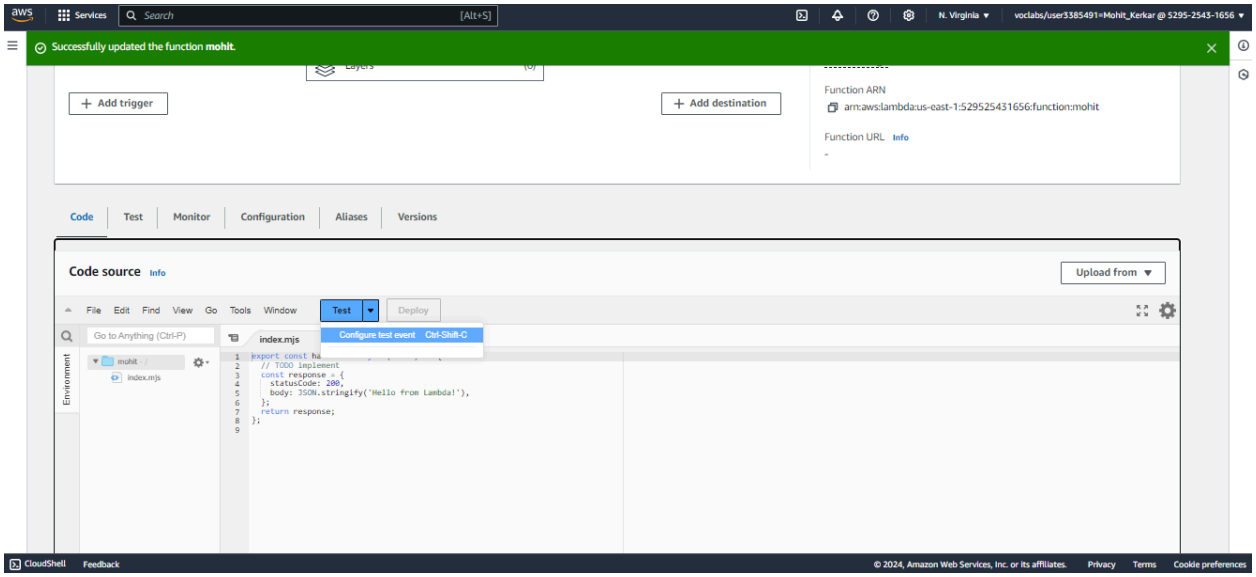
After saving, the configuration setting changes are reflected in the ‘General configuration’ section.



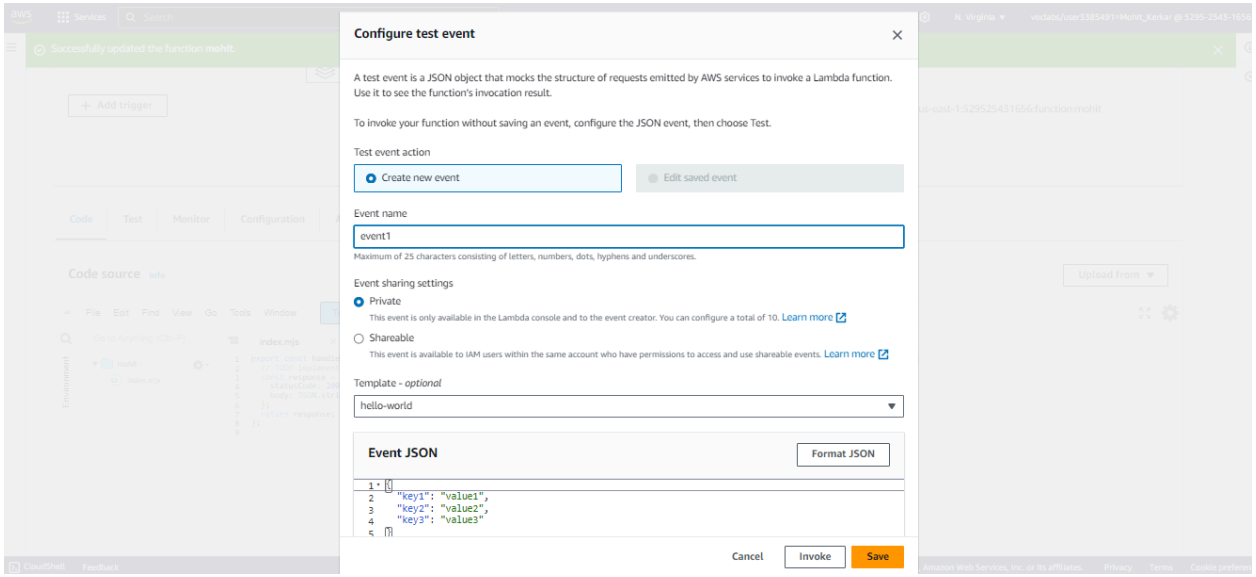
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 3: Creating a new Test event

In our code section, we are given an option to test our code. For this purpose, we are offered numerous test events with which we can possibly test our code.



Click on ‘configure test event’ option for making a new event. Give a suitable name to the event and let other settings be as they are…

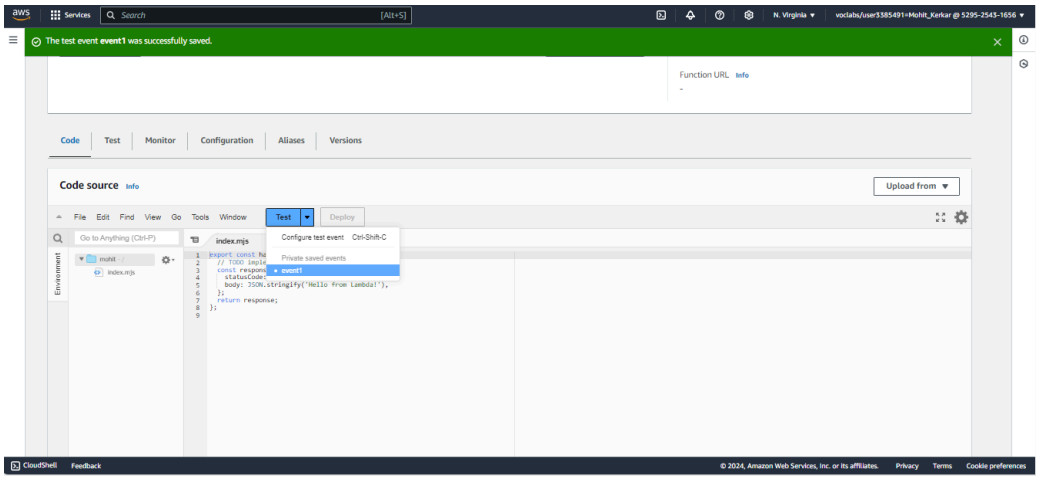


Click on Save..

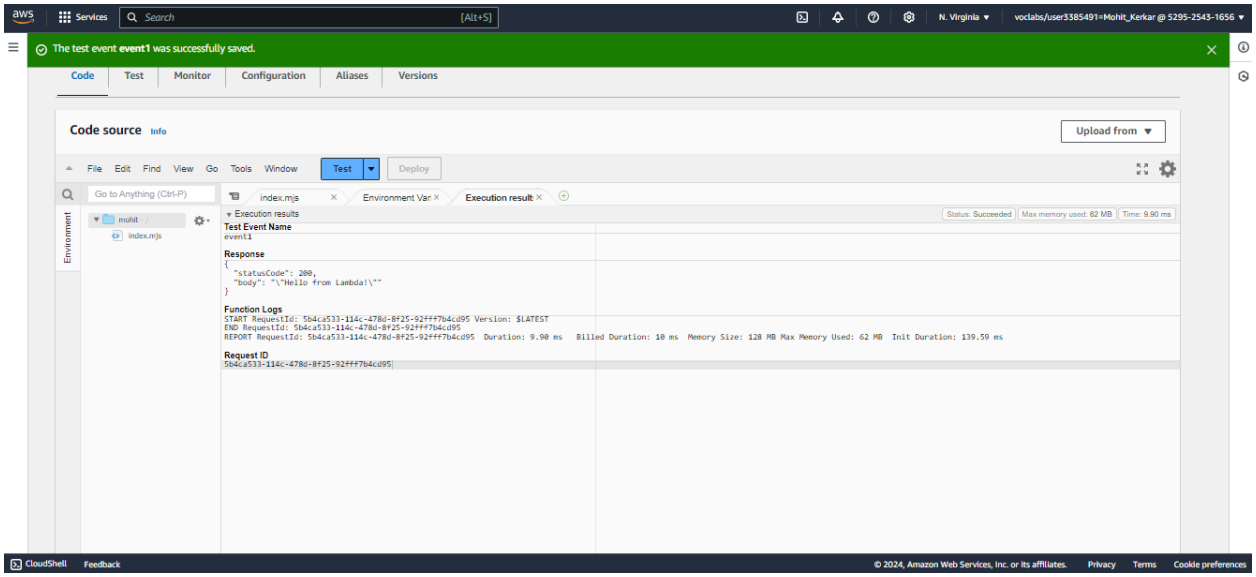
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 4: Test with the new test event that we created

Now, in order to test the code according to the test event that we built in the previous step, again select the Test option and within we will be able to see that test event's name (for me it is event1). Select it and observe the changes that happen



There will be another tab opening which gives us our result after execution. Following is the format with which our Test result comes…

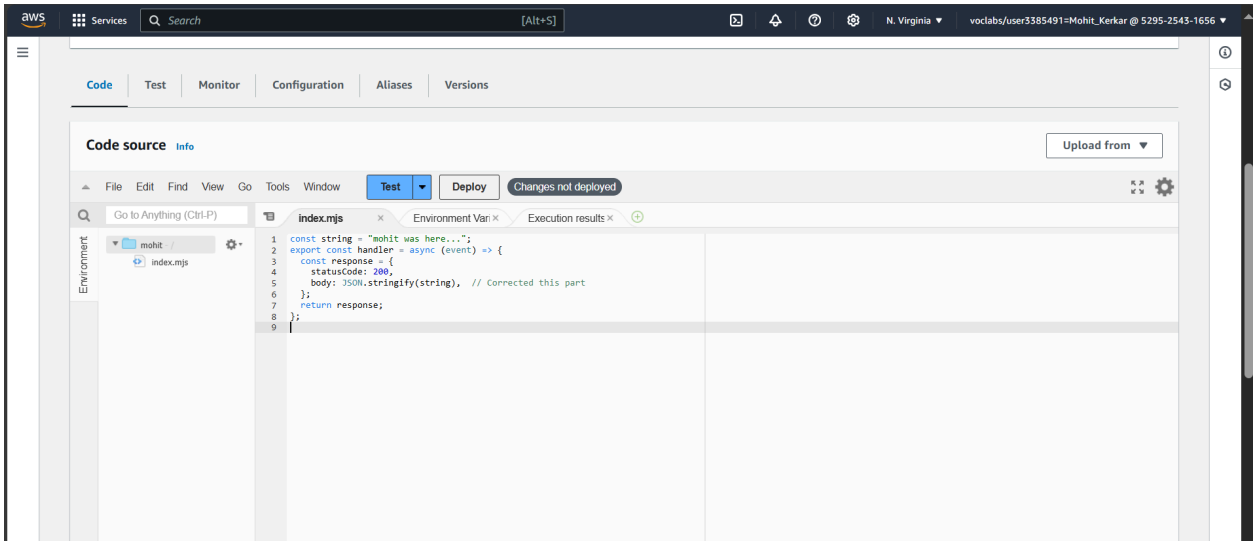


Name: Mohit Kerkar Div: D15C Roll No: 23

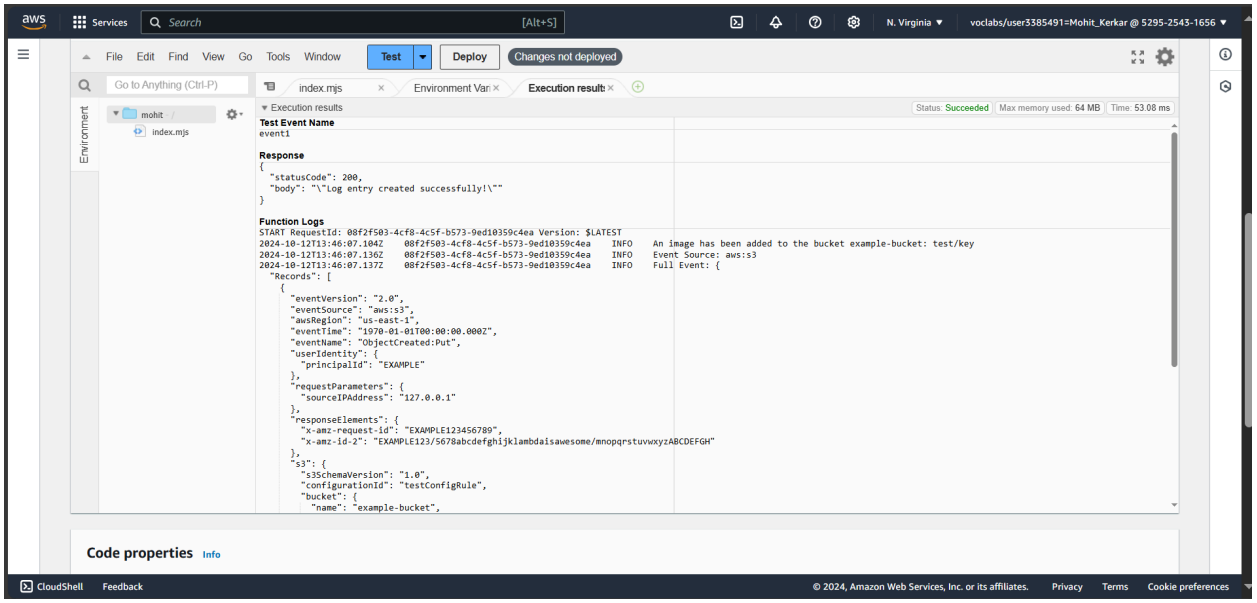
Step 5: Testing our modified code

Now, i added a new line at the start of my index.mjs file in my lambda function

Const string=”mohit was here…”



The execution result displays that string result in the output as well ..



Conclusion: In conclusion, AWS Lambda provides an efficient, serverless computing service that enables you to run code without managing infrastructure. By creating and configuring a Lambda function, you can execute code in response to various events, such as S3 uploads or API requests. The setup process involves defining roles, setting up test events, and modifying the function’s configuration for specific behaviors like timeouts. AWS Lambda’s stateless nature and automatic scaling make it ideal for short-lived, event-driven tasks, such as building APIs or processing data streams, ensuring cost-effective and scalable solutions.